Properties of Artifact Representations for Evolutionary Design

Gregory S. Hornby

QSS Group Inc., NASA Ames Research Center
Mail Stop 269-3, Moffett Field, CA 94035-1000
hornby@email.arc.nasa.gov

Abstract

To achieve evolutionary design systems that scale to the levels achieved by man-made artifacts we can look to their characteristics of modularity, hierarchy and regularity to guide us. For this we focus on design representations, since they strongly determine the ability of evolutionary design systems to evolve artifacts with these characteristics. We identify three properties of design representations – combination, control-flow and abstraction – and discuss how they relate to hierarchy, modularity and regularity.

Introduction

One of the foremost challenges of evolutionary design is improving functional scalability to the level of artifacts designed by people. To achieve human-level complexity of design it is worth looking at how people design complex artifacts. From the fields of software development and engineering, we see that complex artifacts are achieved by exploiting the principles of hierarchy, modularity, and regularity (12) (15) (8).

If we assume that the principles of hierarchy, modularity and regularity are necessary to achieve scalability then our next goal is to design an evolutionary design system capable of producing designs with these characteristics. Since its inception, the field of evolutionary computation has focused primarily on the method for managing the population of solutions, that is, the search algorithm. But the characteristics of evolved designs are limited and biased by the representations that are used to encode them and there has been a growing interest in better representations for evolutionary design. To help in creating design representations capable of producing designs with modularity, hierarchy and regularity it is useful to know which properties representations can have.

Previous Work

Before presenting our own list of properties of design representations we first review related work on classes of design representations and different properties which they can have.

Angeline (3) classifies representations as: translative development functions, mappings that are one-to-one and independent; generative development functions, those with a recursive definition or some type of production system; and adaptive development functions, ones in which the the developmental function for the entire population can change over the course of evolution. This can be summarized as the level to which the developmental system of the representation is applied or evolved: no development, individual-level developmental system, and population-level developmental system.

To better help us create a design representation Bentley and Kumar (4) draw inspiration from embryology to classify representations as:

- No embryogeny, a representation in which there is a one-to-one mapping between elements in the encoded design and elements in the actual design.
- External embryogeny, the developmental rules are not changeable by the search algorithms, rather parameters for such a system are evolved.
- Explicit embryogeny, the rules for creating a design are procedural, such as with genetic programming (10) and Lindenmayer systems (13) but could also be a derivation tree for a grammar.
- Implicit embryogeny, the rules indirectly specify a design, such as with cellular automatas and artificial DNA systems.

In their study they conclude that implicit embryogenies are better than explicit, but their explicit

embryogenies do not include sub-procedural elements such as the automatically defined functions of genetic programming which have been shown to significantly improve the scalability of an evolutionary algorithm (10).

In another comparison Komosinksi and Rotaru-Varga (9) compare three different representations on three different problems within the class of articulated creatures: *simul*, a direct low-level encoding; *recur*, an indirect representation; and *devel*, an indirect developmental encoding. The characteristics they identify for these representations (rated in terms of none/low/variable/high) are: genotype complexity; interpretation complexity; body constraints; brain constraints; modularity; compression; and redundancy. Of these modularity, compression and redundancy are generalizable to other substrate domains whereas the others are domain specific. From their comparison the authors concluded that representations should be high-level and structured.

Stanley and Miikkulainen list five dimensions for artificial embryogenies: cell fate, targeting, heterochrony, canalization, and complexification (14). With this classification rather than either having or not having a property each property is a dimension along which a representation lies. Interestingly, the authors note that the previously reviewed classifications tend to distinguish only between those representations that have reuse from those that do not have reuse but their own classification scheme does not make such a distinction. In addition, while the authors do not make any claims as to which category is most scalable they place natural evolution on the chart for reference as to which properties a representation should have. They score natural evolution as: cell fate has many determination methods; targeting is in between specific and relative; heterochrony is highly flexible; canalization has a high tolerance for imprecision; and for complexification it has a highly variable genotype.

Aside from Angeline's classification, which mainly differentiates between levels in which the development system is applied, these investigations into design representations are somewhat contradictory in their findings. While Bentley and Kumar seem to agree with Stanley and Miikkulainen that representations should be implicit and more like nature, their methods for classifying representations are different. In contrast, Komsinksi and Rotaru-Varga find that representations should be more structured and high-level. None of these investigations give clear guidance on how to construct a better representation.

Programming Languages as a Model

To give better direction for the construction of scalable evolutionary design representations, more precise and more useful definitions of design properties are needed. Here, to develop a classification we identify three properties of representations. For this we use the metaphor of design representations as a kind of computer programming language to define the following features of design representations (7):

- **Combination**: The ability to hierarchically create more powerful expressions from simpler ones. While the subroutines of GLib (2) and genetic programming (GP) (10) allow explicit combinations of expressions, combination is not fully enabled by mere adjacency or proximity in the strings utilized by typical representations in genetic algorithms.
- **Control-flow**: All programming languages have some form of control of execution which permits the conditional and repetitive use of structures. Two types of control-flow are:
  - **Conditionals** can be implemented with an `if`-statement, as in GP, or a condition which governs the appropriate rule to apply, such as in L-systems (5) or cellular automata.
  - **Iteration** is a looping ability, such as the `for`-loop in C/C++ programs.
- **Abstraction**: Has two components to it.
  - **Labeled procedures:** This consists of the ability to encapsulate a group of expressions in the language and label them, enabling them to be manipulated and referenced as a unit. An example of abstraction is the automatically defined functions (ADFs) of GP (10).
  - **Parameters:** The ability to pass parameters to procedures, such as with ADFs or the production rules in parametric L-systems.

In implementation, these elements can be parceled out to different mechanisms, such as branching, variables, bindings, recursive calls, but are nonetheless present in some form in all programmable systems. Some of these basic properties have also been shown to have analogues in biological systems: phenotypes are specified by combinations of genes; the

expression of one gene can be turned on/off by the expression of another gene (11); and an upstream protein can control a downstream protein's activity through a signaling pathway (1).

Properties to Characteristics

Having identified properties of design representations we can now go back to characteristics of human designed artifacts to determine how these characteristics can be achieved.

- hierarchy: the ability to make hierarchical constructs is determined by the ability of a representation to do *combination* on elements of the genotype.
- modularity: can be realized by *abstraction*, which is an encapsulation of a section of the genotype.
- regularity: is achieved through either *iteration* or *abstraction*. Representations which can reuse parts of the genotype to create the phenotype are generative representations (5).

Not mentioned in the above mapping from representational properties to design characteristics are the properties of conditionals or parameters. One use of these two properties is in evolving families of designs (6).

Summary

We can work toward better representations by using the characteristics of hierarchy, modularity and regularity found in man-made design to guide us. In this paper I have borrowed from the field of computer programming languages to identify three properties of design representation – combination, control-flow and abstraction – and matched these properties to the three characteristics of man-made designs. Only be creating representations which have these properties can we hope to produce evolutionary design systems that are able to scale to complex, human-level designs.

References

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of The Cell.* Garland Publishing, New York, 4th edition, 2002.

[2] P. Angeline and J. B. Pollack. Coevolving high-level representations. In C. Langton, editor, *Proceedings of the Third Workshop on Artificial Life*, pages 55–71, Reading, MA, 1994. Addison-Wesley.

[3] P. J. Angeline. Morphogenic evolutionary computations: Introduction, issues and examples. In J. McDonnell, B. Reynolds, and D. Fogel, editors, *Proc. of the Fourth Annual Conf. on Evolutionary Programming*, pages 387–401. MIT Press, 1995.

[4] P. Bentley and S. Kumar. Three ways to grow designs: A comparison of embryogenies of an evolutionary design problem. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Genetic and Evolutionary Computation Conference*, pages 35–43. Morgan Kaufmann, 1999.

[5] G. S. Hornby. *Generative Representations for Evolutionary Design Automation.* PhD thesis, Michtom School of Computer Science, Brandeis University, Waltham, MA, 2003.

[6] G. S. Hornby. Generative representations for evolving families of designs. In E. Cantu-Paz et al., editor, *Proc. of the Genetic and Evolutionary Computation Conference*, LNCS 2724, pages 1678–1689, Berlin, 2003. Springer-Verlag.

[7] G. S. Hornby and J. B. Pollack. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3):223–246, 2002.

[8] C. C. Huang and A. Kusiak. Modularity in design of products and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 28(1):66–77, 1998.

[9] M. Komosinski and A. Rotaru-Varga. Comparison of different genotype encodings for simulated 3d agents. *Artificial Life*, 7(4):395–418, 2001.

[10] J. R. Koza. *Genetic Programming: on the programming of computers by means of natural selection.* MIT Press, Cambridge, Mass., 1992.

[11] B. Lewin. *Genes VII.* Oxford University Press, 2000.

[12] B. Meyer. *Object-oriented Software Construction.* Prentice Hall, New York, 1988.

[13] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants.* Springer-Verlag, 1990.

[14] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003.

[15] K. Ulrich and K. Tung. Fundamentals of product modularity. *Issues in Design/Manufacture Integration - 1991 American Society of Mechanical Engineers, Design Engineering Division (Publication) DE*, 39:73–79, 1991.